Rapport de la Compétition 1
Classification of Extreme Weather Events

Steve Levesque
Département d'informatique et
de recherche opérationnelle
Université de Montréal
steve.levesque@umontreal.ca

Weiyue Cai
Département d'informatique et
de recherche opérationnelle
Université de Montréal
weiyue.cai@umontreal.ca

Abstract

Le but de ce travail de compétition Kaggle est de trouver le meilleur modèle possible de régression linéraire (et le meilleur en général pour la question suivant cela) capable de classifier des événements météorologiques en 3 catégories : standard (0), Cyclones tropicaux (1) et Rivières athmosphériques (2).

NB : Les scores sont biasés puisque par manque de réflection, les données de train et test ont été jumelées pour l'entraînement. Prévoir environ -0.05 à la baisse pour tous les résultats.

1 Feature Design

Nous avons pré-traité les données de plusieurs manières pour entreprendre nos prédictions :

- Standardisation des données.

- Séparation one-hot des données par saison ('spring', 'summer', 'automn', 'winter') après avoir effectué la visualisation des données.

- Choix des attributs inutiles ou répétitifs ('S.No', 'time', 'PS') à exclure du jeu de données.

- Suppression des attributs ('PRECT', 'T200') qui ne sont pas significatifs grâce à la méthode de "feature importance" des modèles random forest, gradient boosting et XGBoost.

Veuillez voir les notebooks en annexe pour plus de détails.

2 Algorithmes

2.1 Régression logistique

Pour les aspects d'algorithmes d'apprentissage, nous avons utilisé la descente du gradient régularisé pour ce qui est de l'optimisation du coût.

2.2 Comparaison des classifiers

Voici la liste des algorithmes qu'on a essayées avec très peu de hyperparameters tuning (avec la moyenne et l'écart-type du taux de précision).

```
Logistic Regression: 0.821001, 0.0034
SVM: 0.844116, 0.0033
Decision Tree Classifier: 0.855653, 0.0025
```

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

```
KNN: 0.889175, 0.0027
Random Forest Classifier: 0.859883, 0.0029
Gradient Boosting Classifier: 0.863714, 0.0027
Extra Trees Classifier: 0.861642, 0.0017
Light Gradient Boosting Machine: 0.889363, 0.0031
eXtreme Gradient Boosting: 0.852931, 0.0034
```

2.3 Réseau des neurones

Nous avons implémenté à l'aide de Keras le réseau des neurones avec 1 hidden layer de 14 neurones. Nous avons appliqué la régularisation L2 sur le output layer.

2.4 Voting Classifier

Nous avons gardé tous les attributs sauf 'S.No', 'time', 'PS' et choisi tous les modèles mentionnés ci-haut sauf KNN pour le soft voting classifier.

2.5 Stacking Classifier

Nous avons choisi deux algorithmes moins performants (XGB et GB) de la famille de boosting, SVM et Random Forest comme la combinaison des estimateurs et Logistic Restression comme l'estimateur final.

3 Méthodologie

3.1 Stratégies de régularisation

En ce qui concerne la régression logistique à partir de zéro, nous avons utilisé des algorithmes gloutons pour arriver à complétion de nos tâches. Par exemple, la fouille pour trouver les meilleurs hyperparamètres est un algorithme "brute-force" glouton.

3.2 Choix des modèles et hyperparamètres

Au niveau du choix des modèles pour le soft voting classifier et le stacking classifier, nous n'avons pas choisi les modèles les plus performants (KNN et Light Gradient Boosting Machine). Certaines raisons peuvent justifier notre choix:

- Très bonne performance sur le modèle d'entraînement pourrait causer le overfitting sur le jeu de données de test.

- Puisque les labels ne sont pas distribués de façon équilibré (le nombre de la classe 0 est beaucoup plus élevé que les deux autres classes), un meilleur taux de précision pourrait s'expliquer par le fait que le modèle prédit plus de 0.

Similairement, nous avons utilisé le GridSearchCV pour trouver les meilleurs hyperparamètres sur le jeu de données d'entraînement. Mais cela ne signifie pas nécessairement que ces hyperparamètres vont donner une meilleur prédiction.

4 Résultats

Voici les résultats des approches qu'on a utilisées avec meilleur score public et privé :

4.1 Régression logistique

4.1.1 Sigmoid

Le meilleur private score: 0.76311 (battu logreg baseline)
public score: 0.75109
Pré-traiement des données:

Table 1: Résultats des Algorithmes sur le Leaderboard

Algorithmes	Public	Private
Régression logistique à partir de zéro avec sigmoid	∼0.75109	∼0.76311
Régression logistique à partir de zéro avec softmax	∼0.75710	∼0.77021
Réseau de neurones	∼0.77131	∼0.78333
Voting classifier	∼0.76721	∼0.77568
Stacking classifier	∼0.78661	∼**0.79043**

- Standardiser les données
- Enlever l'attribut PS
Paramètres:
alpha = 1.5, max iter: 25000, lambda: 3.5.

4.1.2 Softmax

Le meilleur private score: 0.77021 (battu TA baseline)
public score: 0.75710
Pré-traiement des données:
- Standardiser les données - Enlever les attributs PS et PRECT
Paramètres:
alpha = 0.85, max iter = 4000, lambda = 3.

4.2 Autres modèles (avec modules)

4.2.1 Réseau des neurones

Voici les paramètres qu'on a utilisés pour obtenir le meilleur score avec le réseau des neurones.
Private score: 0.78005, Public score 0.77459

```
# create model
model = Sequential()
model.add(Dense(14, input_dim=19, activation='relu'))
model.add(Dense(3, activation='softmax', kernel_regularizer='l2'))
# Compile model
model.compile(loss='categorical_crossentropy',
optimizer='adam',
metrics=['accuracy'])

training_result = model.fit(X_train_nn,
                            y_train_nn,
                            batch_size=10,
                            epochs=50,
                            validation_data=(X_val_nn, y_val_nn))
```

4.2.2 Soft Voting Classifier

Nous avons utilisé le model suivant et toutes les données en enlevant les attributs "S.NO", "PS" et "time".
Private score: 0.77568, Public score 0.76721

```
    final_models = [
          ('lr', LogisticRegression(solver='newton-cg')),
          ('dt', DecisionTreeClassifier()),
          ('rf', RandomForestClassifier()),
          ('gb', GradientBoostingClassifier()),
```

```
        ('etc', ExtraTreesClassifier()),
        ('svc', SVC(probability=True)),
        ('lgbm', LGBMClassifier()),
        ('xgb', XGBClassifier())
        ]
```

4.2.3 Stacking Classifier

Voici le modèle qui nous a donné le meilleur socre:
Private score: 0.79043, Public score 0.78661

```
    models = [
          ('xgb',  XGBClassifier(
                    objective='multi:softprob',
                    eval_metric='merror',
                    colsample_bytree=1,
                    learning_rate=0.02,
                    max_depth=4,
                    n_estimators=10,
                    )
          ),
          ('gb',  GradientBoostingClassifier()),
          ('svc', SVC(C=0.1, random_state=42)),
          ('rf', RandomForestClassifier(max_depth=7))
          ]
    clf_lr = StackingClassifier(
    estimators=models,
    final_estimator=LogisticRegression(C=0.1,
    multi_class='multinomial',
    solver='lbfgs')
)
```

Nous avons fait plusieurs tentatives avec différentes combinaisons des estimateurs et 4 choix de l'estimateur final (Logistic Regression, Decision Tree, Random Forest et LGBM). Le private score est généralement plus grand que 0.77. Avec la combinaison de svm + rf + xgb et gb comme estimateurs et la regression logistique comme l'estimateur final, le private score est toujours plus grand que 0.78.

5 Discussion

5.1 Régression Logistique à partir de zéro

Notre approche tiens compte de méthodes traditionnelles pour faire sa tâche de manière efficace et concise. Voici les avantages et inconvénients :

5.1.1 Avantages

Convivial à implémenter, comprendre, modifier et à accomplir sa tâche de classification.

5.1.2 Inconvénients

Aucune méthodologie poussée utilisée (i.e. Stochastic GD/BFGS, optimal step search) puisque le dernier "baseline" a été battu sans réflexion plus avancées et avec des opérations gloutones. Donc, le modèle souffre très possiblement d'un temps de calcul plus haut que ce qu'il en vaudrait ainsi que d'un temps d'opération (pour compléter la tâche) plus haut.

5.2 Réseau des neurones

5.2.1 Avantages

En général, le réseau de neurones est très performant si on a bien choisi le nombre des layers, le nombre des neurones et les hyperparamètres pour la régularisation et dropout.

5.2.2 Inconvénients

Le réseau des neurones n'est pas nécessairement une approche pertinente pour notre tâche de classification. En plus, les résultats se fluctuent beaucoup même avec les mêmes données, hyperparamètres et machines.

5.3 Voting Classifier

Cette méthode considère tout les algorithmes pour réduire l'"overfitting" et avoir un plus petit écart-type. Il y a deux manière de l'utiliser : soft et hard.

Hard : prend en considération un vote majoritaire sans poids.

Soft : ce cas permet de se baser sur une probabilité pour sélectionner la meilleur classe.

Après l'essai des deux types, le "soft voting" performe mieux.

5.3.1 Avantages

- Peut prendre en entrée une grande sorte d'algorithmes pour les 2 types de voting et permet de bien généraliser la plupart du temps.

5.3.2 Inconvénients

- Un ou plusieurs algorithmes peuvent donner une influence qui falsifie les résultats si on ne fait pas attention aux choix de ceux-ci (i.e. on ne prends pas assez d'algorithmes, on en prends des trop similaires, etc.).

5.4 Stacking Classifier

Il est possible d'empiler des algorithmes, ici appelés des estimateurs pour ensuite passer le résultat à l'estimateur final qui procède à l'entraînement final.

5.4.1 Avantages

Il est possible d'avoir de meilleur résultat et un temps d'entraînenment moindre que Voting Classifier dans le cas où nous avons bien choisi et "tuned" nos algorithmes (estimateurs).

5.4.2 Inconvénients

- Avec une connaissance moins développée des algorithmes (estimateurs) utilisés pour la Stacking, il peut y avoir des difficultés à choisir le groupe d'estimateur et l'estimateur final pour maximiser le résultat.

References

`https://www.coursera.org/learn/machine-learning`

ufldl tutorial Softmax Regression, `http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/`

Logistic and Softmax Regression, `https://houxianxu.github.io/2015/04/23/logistic-softmax-regression/`

Softmax Regression, `https://zhuanlan.zhihu.com/p/98061179?utm_source=wechat_session&utm_medium=social&utm_oi=774718892074061824`

https://github.com/hankcs/CS224n/tree/master/assignment1

https://github.com/hartikainen/stanford-cs224n/tree/master/assignment1

Scikit-Learn Stacking Classifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

Scikit-Learn Voting Classifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

Keras API reference, https://keras.io/api/

XGBoost Parameters, https://xgboost.readthedocs.io/en/latest/parameter.html